CSL303 – Introduction to Object Oriented Methodology

Lab Assignment – 3

March 2021

Submission Date: March 29, 2021 11.00 PM

Programming Language: **Java**

1. Design a class `Person` which has `name, address, phoneNo` and `emailId.` Design two abstract classes `Student` and `Employee` which are subclasses of the `Person` class.

- The `Student` class has `basic_fee, other_fee` and `status` as data members. "`status`" data member takes values as `Freshman, Sophomore, Junior, Senior`. (if the student is in 1st year, 2nd year, 3rd year and 4th year of graduation respectively). The `Student` class has an abstract method `calculateFee().` Two classes `Sponsored` and `NonSponsored` are derived from the `Student` class which displays the details of the `Student` as well as the `fees` of the student.
  `Fees` is calculated as:
  - For `Sponsored student:` Total fee = 100% of basic fee + status * (other_fee).

  Ex: If the student is sponsored and is a Freshman: *Total fee =100% of basic fee+1*(other_fee)*

  - For `NonSponsored student`: *Total fee = 70% of basic fee + other fee*

  Ex : if the student is a `NonSponsored` one: Total fee = 70% of basic fee + other fee

- The class `Employee` has 2 abstract methods `calculateIncrement()` and `calculatePromotedLevel().` The classes `Faculty` and `Staff` are inherited from `Employee` class. These two classes have `Salary`, `level, working_hours` data members.
  `calculateIncrement` is calculated as:
  Faculty : Increment = 125% of salary
  Staff: Increment = 110% of salary
  `calculatePromotedLevel` is calculated as:
  - `Faculty`: If no of working hours are greater than 1000, then level is incremented by 1, else there is no change in the level.
  - `Staff`: If no. of working hours are greater than 500, then level is incremented by 1, else there is no change in the level.

Instantiate the classes `Sponsored, NonSponsored` and display their details along with fees. Also display the details of the faculty and staff along with the promoted level and increment by instantiating both the classes.

2. Design a class `Account` with data members `Account_no`, `balance`, `available_loan_limit`, `outstanding_loan_amount`, `amount_paid_monthly` (array that stores amount repaid for each month so far), `no_of_credit_applications` (number of times the account holder has applied for loan), `no_of_times_accepted`.

   Derive a class `CIBIL` which has a method `calculateCIBIL()` and `isEligible()`. `calculateCIBIL()` will calculate the `CIBIL` score of a particular account and display the `CIBIL` score whose value is less than 1000. `CIBIL` score can be calculated based on the following factors:

   1. Credit history: This contains the record of past 5 months and has 40% weightage in the formula for calculating `CIBIL` score.
   2. Credit Utilization: This can be calculated as current outstanding loan amount divided by the available balance. It has a weightage of 35%.
   3. Credit applications: No. of times loan got accepted divided by No. of times the account holder has submitted credit application. Weightage is 25%.

   CIBIL = Credit history + credit utilization + no. of credit applications.

   The function `isEligible()` returns true if the `CIBIL` score is greater than 600. Returns false if it is less than 600.

   Example: Account no.: 1234

   Balance : 50000

   Available_loan_limit : 500000

   outstanding_loan_amount : 400000

   amount_paid_monthly : 20,000 | 20,000 | 20,000| 0 |20,000

   no_of_credit_applications : 3

   no.of_times_accepted : 2

   CIBIL SCORE CALCULATION

   Credit history: 30% of 1000 *(4/5) [4 one month payment has been missed]

   Credit Utilization : 35% of 1000 *(400000/500000)

   credit_applications: 25% of 1000 * (2/3)

   CIBIL = 240 + 280 + 167 = 687

   IsEligible = true

3.